VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Absolvování individuální odborné praxe
Individual Professional Practise in the Company

2010 Lukáš Kuděla

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 5. května 2010 …….

podpis

Abstrakt

Práce pojednává o průběhu studentské praxe ve firmě Tomas Ivansky, RXM Research.com. Nejdříve tuto firmu velmi krátce popíši a vysvětlím své umístění a úlohu ve firmě během praxe. Dále uvádím seznam úkolu, které mi byly uděleny. První úkoly byly více méně pomocné práce, díky kterým jsem se seznámil se strukturou databáze banky. Další úkoly se už zabývali vývojem a analýzou software. Neopomenu přínos, kterým pro mne praxe byla, zmíním také uplatněné znalosti získané během studia a zároveň scházející dovednosti.

Klíčová slova

Banka, Databáze, Analýza, Vývoj software, Java, Use case, E-R diagram

Abstract

The topic of this article is my student s practice in Tomas Ivansky, RXM Research.com. At first I shortly describe this company and my place and task in this company during my practice. Next is a list of tasks that I have been granted. The first tasks were more or less auxiliary tasks, thanks to whom I became acquainted with the structure of the database of the bank. Another task is already engaged in the development and analysis software. I mention the contribution of the praxis for me, I also mention my knowledge used by study and also missing craft.

Key words

The bank, Database, Analysis, Software development, Java, Use case, E-R diagram

Použité zkratky

IT	Information technology
E-R	Entity-Relationship
UML	Unified Modeling Language
SQL	Structured Query Language
CASE	Computer Aided Software Engineering

Obsah

1 Popis odborného zaměření firmy

Firma Tomas Ivansky, RXM Research.com byla založena jako entita s minimem režijních aktivit, zaměřená na následující služby:

- Návrh a optimalizace transakčních bankovních informačních systémů.
- Expertní služby v oblasti elektronického vypořádání plateb, clearingu a eliminace souvisejících rizik.
- Expertní služby v oblasti transakčních schémat hromadného elektronického účtování.
- Forezní aktivity.

Firma tyto služby poskytuje jak tuzemským, tak zahraničním bankovním subjektům a spolupracuje ve výše uvedených oblastech s významnými auditorskými a konzultantskými společnostmi.

2 Popis pracovního zařazení

V této firmě jsem se podílel na vývoji nového systému pro jednu nejmenovanou banku (dále jen banka) jako IT analytik, i když jeden úkolů zahrnoval z větší části programování.

IT analytik analyzuje požadavky procesů a potřeb, a podle toho navrhuje schematické diagramy částí informačních systémů a jejich celků. Analýza se zabývá nejen technickým provedením, ale také funkčními požadavky a legislativními podmínkami. Ke každému projektu vzniká i podrobná technická dokumentace včetně datových struktur, nadefinovaných rozhraní a důležitých výkonnostních testů.

3 Seznam úkolů v průběhu odborné praxe

3.1 Úprava diagramů

3.1.1 Zadání

Banka vyměnila v září CASE nástroj Enterprise Architect od Sparx Systems za PowerDesigner od Sybase. CASE nástroje primárně umožňují, modelování IT systému pomocí diagramů (člověk lépe chápe obrázek než složitě psané slovo), generování zdrojového kódu z modelu (usnadňuje práci programátorům), zpětné vytvoření modelu podle existujícího zdrojového kódu (reverse engeneering), synchronizaci modelu a zdrojového kódu, vytvoření dokumentace z modelu. Přestože byl vyvinut a akceptován v rámci UML 2.0 standard, definovaný standardizační skupinou Object Management Group, pro výměnu metadat mezi různými CASE nástroji, většina producentů těchto komerčních nástrojů tento standard zcela nerespektuje. Důsledkem toho je omezená přenositelnost diagramů mezi různými CASE nástroji. Tento problém nastal i tady, kdy E-R diagramy v PowerDesigneru [1] byly zcela nečitelné a nepřehledné. Proto bylo mým prvním úkolem upravení těchto diagramů do přehledné podoby, aby v nich vývojáři mohli snadno a rychle číst.

E-R diagram je diagram vytvořený pomocí metody E-R modelu. Je to metoda popisující uživatelskou aplikaci na konceptuální úrovni abstrakce za účelem návrhu struktury databáze. Obsahuje především jednotlivé typy entit (Entity types) a typy vztahů (Relationship types) - odtud název modelu. *Typ entity* (entitní typ) je jednoznačně profilovaný a nezávisle existující objekt reálného světa. *Typ vztahu* (vztahový typ) je vazba mezi dvěma nebo více typy entit. Samotný pojem entita či vztah označuje výskyt (instanci) typu entity či typu vztahu. Entitám a vztahům lze přiřazovat určitou vlastnost označovanou jako *atribut*. Podle různých hledisek lze rozlišit různé typy atributů:

- identifikační (klíčové) - jsou součástí identifikačního klíče entity,
- popisné - nejsou součástí identifikačního klíče entity,
- atomické (jednoduché) - přiřazují entitnímu nebo vztahovému typu nejvýše jednu, dále již nedělitelnou hodnotu,
- skupinové (složené) - tvořeny několika složkami (např. adresa), umožňují odkazy na celý atribut nebo jeho jednotlivé složky,
- vícehodnotové - mohou obsahovat více rovnocenných hodnot.

Přípustná množina hodnot atributu se nazývá *doména atributu*. Typy entit identifikovatelné pouze svými atributy jsou silné typy entit (regulární). Ostatní typy entit jsou slabé typy entit (pro jednoznačnou identifikaci musí být použit i atribut jiného typu entity - identifikačního vlastníka).

Postup vytváření E-R modelu je obvykle následující:

- identifikace typů entit,

- identifikace typů vztahů,
- definice atributů,
- definice integritních omezení.

UML je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. UML nabízí standardní způsob zápisu jak návrhů systému včetně konceptuálních prvků jako jsou business procesy a systémové funkce, tak konkrétních prvků jako jsou příkazy programovacího jazyka, databázová schémata a znovupoužitelné programové komponenty. Podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů. Neobsahuje způsob, jak se má používat, ani neobsahuje metodiku, jak analyzovat, specifikovat či navrhovat programové systémy.

3.1.2 Řešení

Aby byly diagramy přehledné a čitelné, nesměla se tam žádná entita a vztah překrývat, díky tomu jsem musel entity přesouvat a u některých dokonce skrýt i atributy. Upravení všech diagramů my zabralo celý měsíc.

3.2 Doplnění popisků k atributům a tabulkám

3.2.1 Zadání

Po úpravě diagramu nastal ještě jeden problém, který ale byl už i v Enterprise Architect, a to že u většiny atributů a tabulek chyběly popisky. Základní návrh databáze popisky všech atributů i tabulek obsahoval, ale do Enterprise Architect se už nedostaly. A proto vždy, když vývojáři nevěděli k čemu jaký atribut slouží, museli se podívat do původních dokumentů, kde tyto popisky byly. Tak jsem dostal další úkol, doplnit tyto popisky.

3.2.2 Řešení

Dostal jsem do rukou původní dokumenty k návrhu tabulek a mohl jsem začít přidávat popisky. Byla to nudná a zdlouhavá práce, ale díky ní jsem se podrobněji seznámil se strukturou celé databáze a zjistil, co všechno taková databáze banky musí obsahovat. Samozřejmě některé atributy se za dobu existence databáze zrušily, některé zase přibyly a to stejné platí i u celých tabulek. Ale zrušených tabulek bylo opravdu málo, zato těch, kterých přibylo, bylo mnoho násobně více. Popisky jsem přidával jenom k těm atributům a tabulkám, které se nacházely v původním návrhu, a u těch ostatních jsem neměl jinou možnost, než je nechat prázdné. Díky velikosti databáze mi přidání popisků zabralo něco málo přes měsíc.

3.3 Use case sazebník

3.3.1 Zadání

Dalším z mých úkolů bylo navrhnout use case diagram pro sazebník a následně popsání jednotlivých use case. Use case diagram zachycuje vnější pohled na modelovaný systém, a tím

pomáhá odhalit hranice systému a slouží jako podklad pro odhady rozsahu. Jde o posloupnost souvisejících transakcí mezi účastníkem (zpravidla uživatelem v určité roli, ale také jiným systémem) a systémem během vzájemného dialogu. Hlavním účelem je zachycení aktérů, kteří se systémem komunikují a vztahů mezi službami a těmi, kterým jsou poskytovány, a to vizuální i textovou podobou, která je srozumitelná vývojářům systému i zákazníkům (tj. těm, kteří jej mají používat).

3.3.2 Řešení

Use case diagram jsem navrhl bez větších problémů, díky znalostem z úvodu do softwarového inženýrství. Problém ale nastal při popisu jednotlivých use case, kdy jsem zjistil, že nemám vůbec žádné znalosti s jejich popisem. Naštěstí se mi do rukou dostal dokument popisující psaní těchto use case [2], díky němuž jsem byl schopný jednotlivé use case popsat.

Use case je technika pro zdokumentování případného požadavku na nový systém, nebo změny na stávající systém. Každý use case, poskytuje jeden nebo více scénářů, které zaznamenávají, jak by systém měl spolupracovat s koncovým uživatelem, nebo jiným systémem k dosažení konkrétních hospodářských cílů. Zde se obvykle nepoužívá technický jazyk, je preferován jazyk koncového uživatele nebo doménového expert. Use case jsou ve většině případu vytvářené autorem požadavků ve spolupráci s ostatními zúčastněnými stranami.

Use case jsou jednoduché nástroje na popsání chování softwaru nebo systému. Use case obsahují textový popis všech možný způsobů, jak uživatel může pracovat se softwarem nebo systémem. Use case nepopisují žádné interní procesy systému, nebo jak bude systém implementován. Jednoduše ukazují kroky jak má uživatel provést úkol/aktivitu. Všechny způsoby, kterými uživatelé komunikují se systémem, lze popsat tímto způsobem. Popsání jednotlivých use casu mi trvalo okolo 3 týdnů.

3.4 Přidání oddělovačů

3.4.1 Zadání

V rámci vývoje nového systému pro reportování dat o klientech a úvěrech bylo zapotřebí navrhnout a implementovat procedury a nástroje umožňující komperativní testování stávajícího a nově vyvíjeného systému. K tomu účelu bylo třeba vyvinout nástroj, jehož prostřednictvím budou data produkovaná původním systémem natažena do testovací databáze a zpřístupněna pro komperativní testy. Data produkována původním systémem mají ovšem vážnou vadu, a to že jednotlivé atributy nejsou odděleny žádným oddělovačem a díky tomu nemohou být hromadně nahrána do databáze. Na mě bylo vytvořit aplikaci, která za jednotlivé atributy vloží oddělovač, aby bylo možno tyto data hromadně nahrát do testovací databáze.

3.4.2 Řešení

Pro vytvoření aplikace na přidání oddělovačů jsem využil jazyku Java a nástroje Netbeans, který nabízí práci s uživatelským rozhraním, jelikož nástroj Eclipse, který je používán celou

bankou, nenabízí práci s uživatelským rozhraním. Data produkována původním systémem byla zapsána, tak že každý atribut byl zapsán v jeho maximální znakové délce. Což např. pro atribut, který má max. délku 20 znaků a je v něm záznam obsahující jenom 5 znaků, se do souboru zapsalo těchto 5 znaků a za ně 15 mezer, poté následoval další atribut. Proto jsem si musel nejdříve zjistit maximální délky všech atributů pro správné rozdělení záznamu na jednotlivé atributy. Poté stačilo načítat jednotlivé záznamy, přidávat oddělovače za každý atribut a tyto záznamy s oddělovači zapsat do nového souboru. Oddělovačem jsem zvolil středník. Po vytvoření aplikace. jsem měl tyto data nahrát do testovací databáze. V ní se ale ještě nenacházely tyto tabulky, a tak jsem je vytvořil jednoduchým SQL příkazem

```
CREATE TABLE nazev_tabulky
   (<nazev sloupce> <datovy typ>
     [DEFAULT <konstantni vyraz>]
     [NULL | NOT NULL]
     [<omezeni pro sloupce>]
     |[<omezeni pro tabulku>]
     [,...n]
   )
```

a data do databáze nahrál pomocí SQL příkazu

```
LOAD FROM <nazev_souboru> DELIMITER ";" INSERT INTO <nazev_tabulky>;
```

Vytvoření aplikace mi zabralo jeden týden.

3.5 Návrh formulářů

3.5.1 Zadání

Mým posledním úkolem bylo vytvoření několika návrhů formulářů do core-byznysového systému banky spravující kompletní životní cyklus všech produktů poskytovaných bankou, včetně účtovaní o nich. Na těchto formulářích jsem pracoval s p. Ivanským. Finalizace všech formulářů nám zabrala 4 měsíce.

3.5.2 Řešení

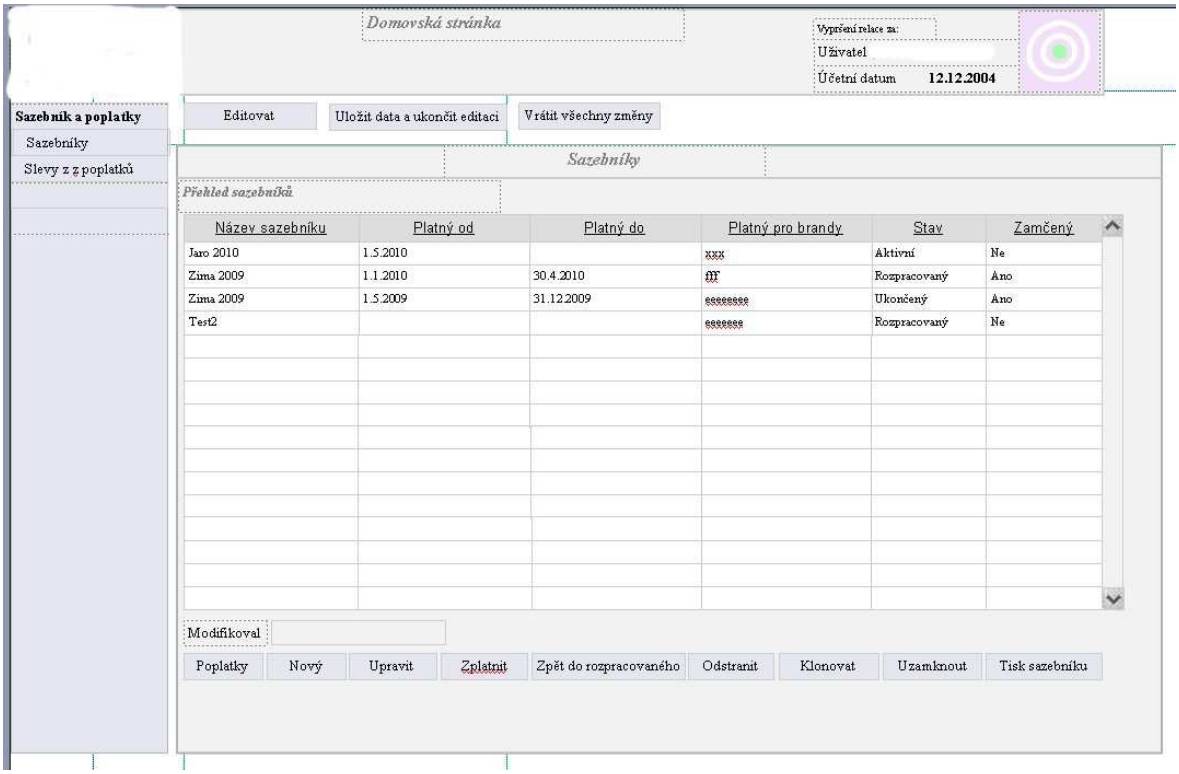
Tyto návrhy měly obsahovat návrh uživatelského rozhraní a popsání jednotlivých formulářů tak, aby nad nimi programátor nemusel moc přemýšlet a mohl je rovnou vytvořit. Návrhy byly pro čtyři formuláře, sazebníky, poplatky, typy slev a úvěry – cenové nastavení úvěru. Pro návrhy uživatelských rozhraní jsem používal Microsoft Office Publisher a vycházel jsem ze základního konceptu formulářů ze stávajícího systému. Pro popsání formuláře mi byl dodán dokument [3], z jehož kostry jsem vycházel. Tato kostra se skládala z:

- návrhu a uspořádání formuláře,
- aktivních prvků formuláře,
- položek formuláře,
- metod formuláře.

3.5.2.1 Formulář Sazebník

Nejdříve jsme tento formulář navrhovali s formulářem poplatky jako jeden, ale od toho se upustilo, jelikož by tak bylo na jednom formuláři až příliš funkcionality a uživatelé by z toho mohli být zmateni. Proto jsme je rozdělili do dvou formulářů.

Tento formulář umožňuje uživateli spravovat sazebníky (viz Obrázek 1).



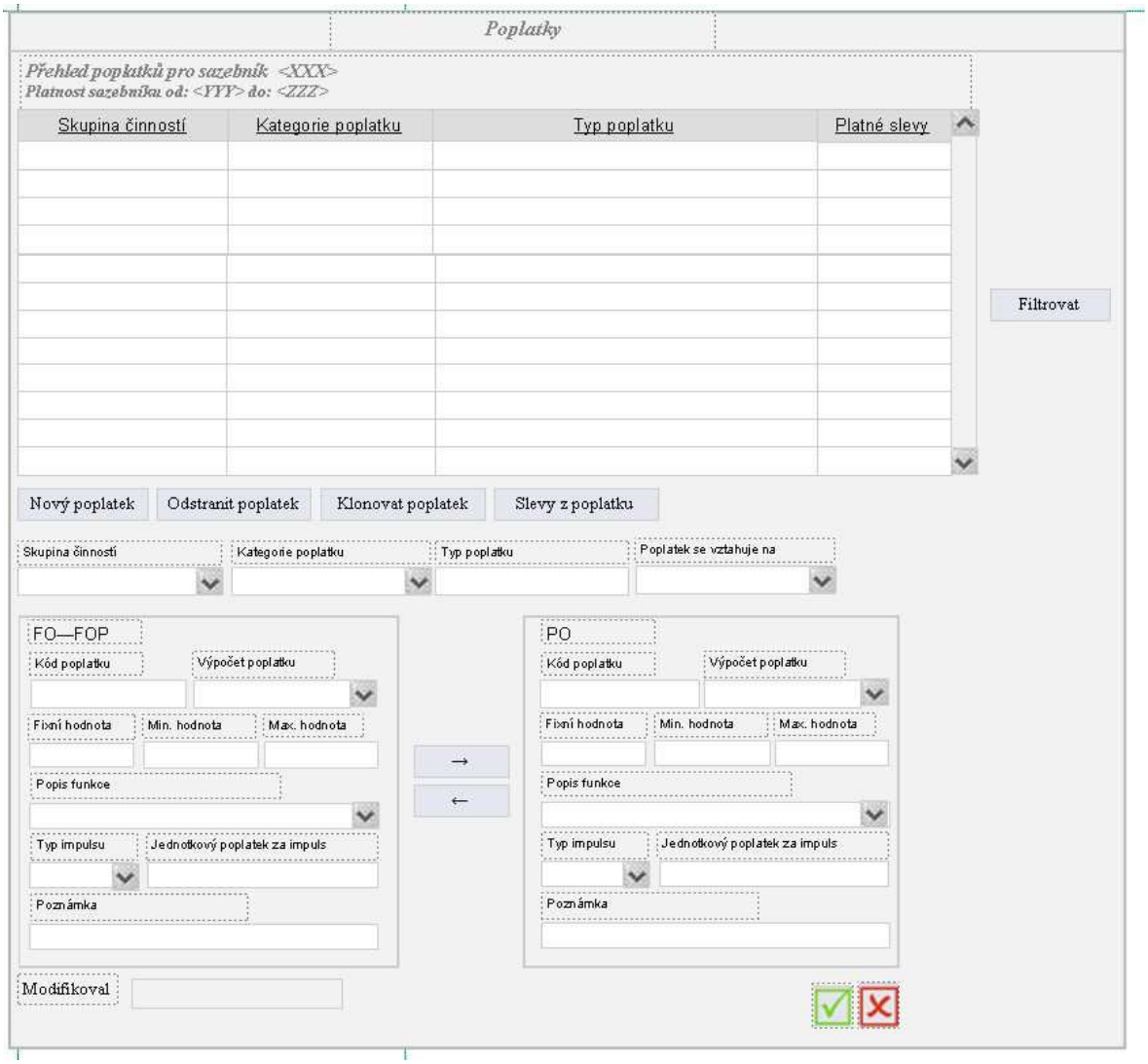
Obrázek 1: Uživatelské rozhraní pro Sazebníky

3.5.2.2 Formulář Poplatky

Tento formulář je volán z modulu Sazebník. Uživatel ve formuláři edituje jednotlivé poplatky spojené se sazebníkem vybraným ve formuláři Sazebník (viz Obrázek 2).

V horní části je přehled všech poplatků, spojených s daným sazebníkem. Spodní část formuláře plní dva účely:

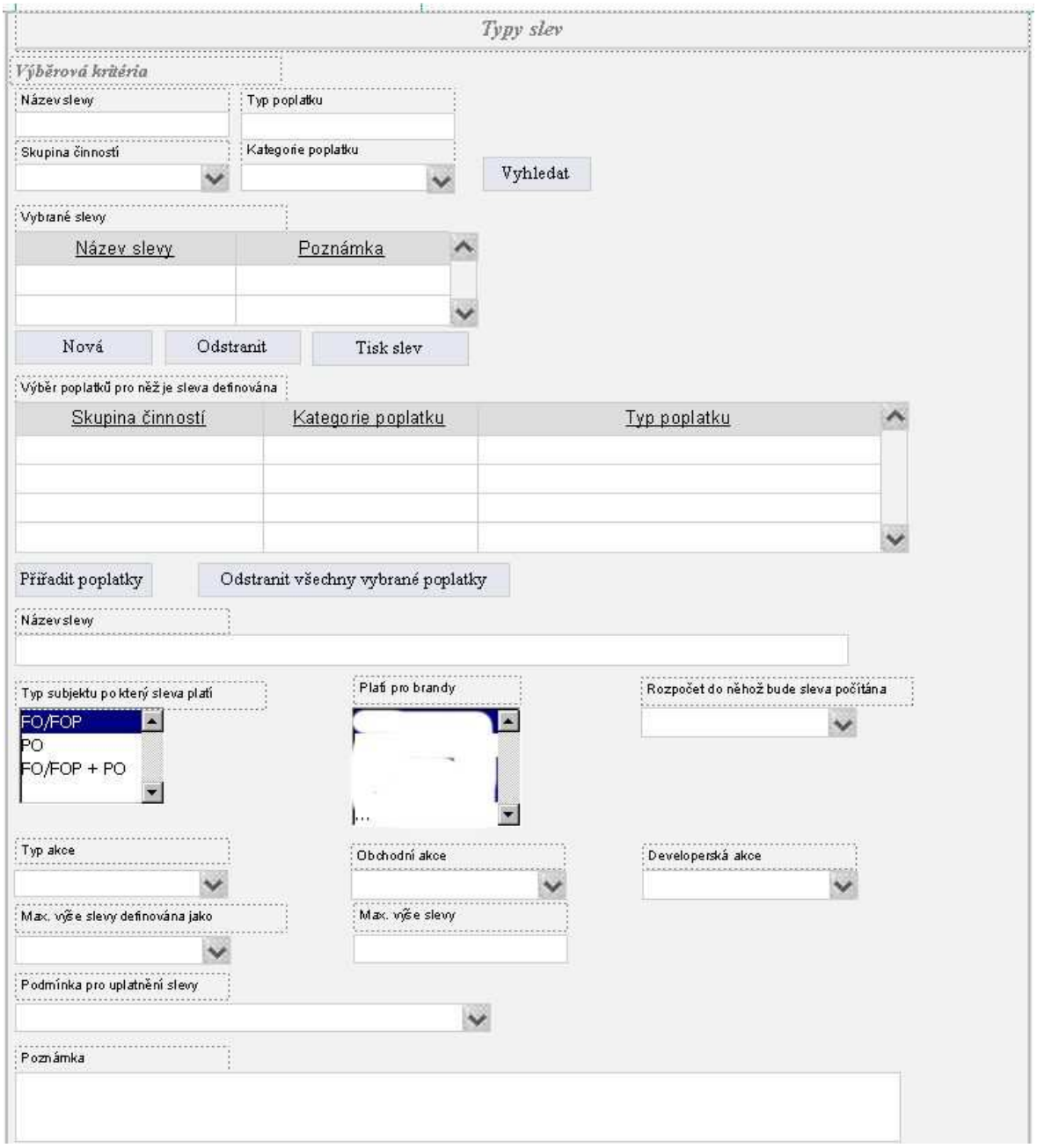
1. Zobrazuje doplňující informace k záznamu, vybranému v horní části, které se do přehledu nevešly.
2. Umožňuje zakládat nové a editovat stávající záznamy položek poplatků.



Obrázek 2: Uživatelské rozhraní pro Poplatky

3.5.2.3 Formulář Typy Slev

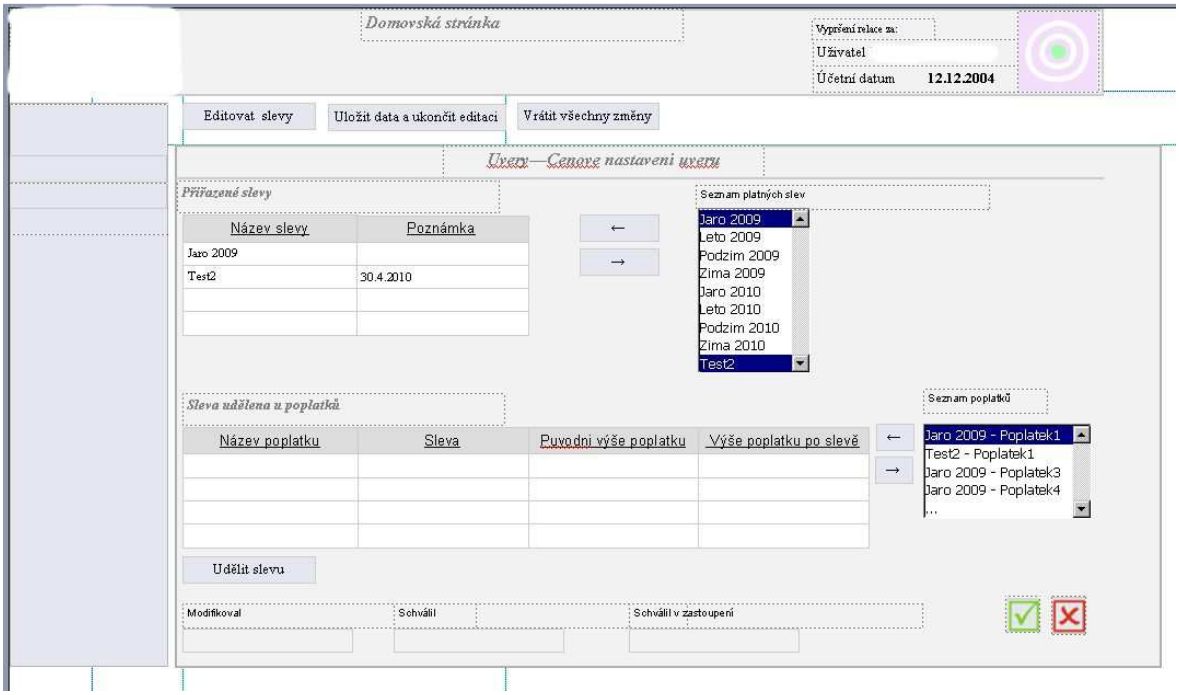
Formulář je volán buď samostatně z hlavního menu, nebo je volán z modulu Sazebník. V tomto formuláři uživatel edituje požadované slevy. Tyto slevy může současně propojovat s množinou poplatků, nad nimiž jsou slevy definovány (viz Obrázek 3).



Obrázek 3: Uživatelské rozhraní pro Typy slev

3.5.2.4 Formulář Úvěry – cenové nastavení úvěru

V horní části formuláře si uživatel vybere ze seznamu platných slev slevy, na jejichž základě jsou vyhledávány poplatky, zobrazující se ve spodní části formuláře. Zde má uživatel možnost přidělení konkrétních slev k poplatkům (viz Obrázek 4).



Obrázek 4: Uživatelské rozhraní pro Úvěry – cenové nastavení úvěru

4 Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe

Během praxe jsem nejvíce uplatnil znalosti získané v těchto předmětech:

1. Databázové a informační systémy,
2. Teorie zpracování dat,
3. Úvod do softwarového inženýrství,
4. Uživatelská rozhraní,
5. Programovací techniky.

5 Znalosti či dovednosti scházející studentovi v průběhu odborné praxe

Nejvíce mi scházela znalost psaní jednotlivých use casu a znalost popisování formulářů. Také jsem neznal nástroj PowerDesigner.

6 Dosažené výsledky a jejich celkové zhodnocení

Praxe ve firmě Tomas Ivansky, RXM Reasearch.com pro mě byla obrovská zkušenost. Zjistil jsem, jak vypadá práce na skutečných projektech, a jak se pracuje v týmu několika lidí. Prohloubil jsem si znalosti v analýze a vývoji software, naučil jsem se pracovat s CASE nástrojem PowerDesigner, poznal jsem, jak vypadá struktura databáze opravdové banky, a co všechno taková databáze musí obsahovat.

7 Použitá literatura

[1] *PowerDesigner15.2*
URL:
< http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.powerdesigner.15.2/doc/html/title.html /> [cit 2.5.2010]

Interní dokumenty podléhající spřízněnému režimu banky:

[2] Rudolf Čermák, *UseCases-WithTemplates-EN*
[3] Tomáš Ivanský, *Formulář Frm FS-002-08 Změna stavu*